基于交叉迁移和共享调整的改进蝴蝶优化算法 *

孙 林[1,2†]，陈岁岁[1]，徐久成[1,2]，王振华[1]

(1. 河南师范大学 计算机与信息工程学院，河南 新乡 453007；2. 河南省高校计算智能与数据挖掘工程技术研究中心，河南 新乡 453007)

摘 要：针对蝴蝶优化（monarch butterfly optimization，MBO）算法易陷入局部最优和收敛速度慢等问题，提出了一种基于改进的交叉迁移和共享调整的蝴蝶优化（MBO with cross migration and sharing adjustment，CSMBO）算法。首先，利用基于维度的垂直交叉操作来替换标准 MBO 算法的迁移算子，形成交叉迁移算子，有效提升其搜索能力；其次，将原始调整算子改为具有信息分享功能的共享调整算子，以加快算法的收敛速度；最后，采用贪婪选择策略取代标准 MBO 算法中的精英保留策略，减少一次排序操作进而提高其计算效率。为了验证 CSMBO 算法的优化能力，测试了其在 30 维和 50 维的函数优化，并与三种优化算法进行比较，其实验结果表明 CSMBO 算法具有良好的优化性能。

关键词：智能优化；蝴蝶优化；交叉迁移；共享调整

中图分类号：TP301.6 doi: 10.3969/j.issn.1001-3695.2018.07.0611

Improved monarch butterfly optimization algorithm based on cross migration and sharing adjustment

Sun Lin[1,2†], Chen Suisui[1], Xu Jiucheng[1,2], Wang Zhenhua[1]

(1. *College of Computer & Information Engineering*, *Henan Normal University*, *Xinxiang Henan* 453007, *China*; 2. *Engineering Technology Research Center for Computing Intelligence & Data Mining of Henan Province*, *Xinxiang Henan* 453007, *China*)

Abstract: In order to solve the problems of the monarch butterfly optimization (MBO) algorithm, such as it is easy to fall into local optimum and the convergence speed is low, this study proposed an improved MBO algorithm with cross migration and sharing adjustment (CSMBO) . Firstly, this paper introduced a dimension-based vertical crossover operation to substitute the original migration operation of MBO, and then generated a cross migration operator. Thus, this operation could improve search ability of MBO algorithm effectively. Secondly, in order to speed up the convergence of MBO algorithm, the sharing adjustment operator with information sharing replaced the original adjustment operator. Finally, this paper utilized the greedy strategy to instead of the elite strategy of MBO, which could reduce one sorting operation and improve the calculation efficiency of MBO algorithm. To evaluate the optimization ability of our CSMBO algorithm, this paper made some experiments on a set of common benchmark functions with 30-dimensions and 50-dimensions, and the results showed that the proposed CSMBO algorithm had good optimization performance, and outperformed currently available three optimization approaches with which it is compared.

Key words: intelligent optimization; monarch butterfly optimization; cross migration; sharing adjustment

0 引言

优化是在某些约束条件下搜索和找到给定问题的最佳解的过程[1]。对于优化算法，它们可以分为确定性算法和随机算法两大类[2]。近年来，许多可以转换成优化问题的现实问题变得越来越复杂，并且很难通过确定性算法来解决[3~5]。因此，在实际应用中对随机算法的研究已成为一个很有前景的热点课题[6,7]。如今，群智能优化算法作为随机算法的一个重要分支，已展示出解决复杂优化问题的强大而有效的能力[8]。其中比较著名的有：粒子群优化算法（particle swarm optimization, PSO）[9~12]、烟花算法[13]、生物地理学优化算法[14]、蝴蝶优化（MBO）算法[15]、萤火虫优化算法[16~18]等。而在这其中，蝴蝶优化算法因其优化性能优异而得到广泛关注。

蝴蝶优化算法是 2015 年由 Wang 等人[15]受自然界黑脉金斑蝶迁徙行为为启发而提出的。MBO 算法有两个重要的算子即迁移算子和调整算子。前者提供一定的局部搜索能力，后者提供一定的全局搜索能力。在 MBO 算法中，蝴蝶的搜索方向由迁移算子和调整算子确定，而这两个算子可以同时执行，因此，MBO 算法很适合用于并行处理，并且能够很好

收稿日期：2018-07-16；**修回日期：**2018-10-31 **基金项目：**国家自然科学基金资助项目（61772176，61402153，11601130）；中国博士后科学基金资助项目（2016M602247）；河南省科技创新人才项目（184100510003）；河南省科技攻关项目（182102210362，182102210078）；河南省高校青年骨干教师培养计划项目（2017GGJS041）；河南省自然科学基金资助项目（182300410130，182300410368）；新乡市科技攻关计划项目（CXGG17002）；河南省高等学校重点科研计划项目（14A520069，17A520038，16A520015）；河南师范大学博士科研启动费支持课题（qd15132）；河南师范大学青年科学基金项目（2015QK23）；国家教育部产学合作协同育人项目（201702115008）

作者简介：孙林（1979-），男（通信作者），河南南阳人，副教授，硕导，博士，主要研究方向为粒计算、智能优化算法、大数据挖掘等（sunlin@htu.edu.cn）；陈岁岁（1993-），女，河南商丘人，硕士研究生，主要研究方向为智能优化算法、数据挖掘等；徐久成（1964-），男，河南洛阳人，教授，博导，博士，主要研究方向为粒计算、大数据挖掘等；王振华（1978-），男，河南新乡人，讲师，硕士，主要研究方向为智能优化算法.

地平衡局部搜索和全局搜索。由于 MBO 算法参数少且易于实现，已经被广泛应用到很多领域，如 0-1 背包问题[19]、自动电压调节[20]、神经网络训练[21]、动态路径问题[22]、骨质疏松症的分类问题[23]等。但 MBO 算法在解决一些复杂问题时仍存在搜索能力较弱等缺陷。因此，许多学者对其进行了改进研究。Wang 等人[24]提出了一种基于自适应交叉算子和贪婪策略的 MBO（monarch butterfly optimization with greedy strategy and self-adaptive crossover operator, GCMBO）算法，提高了 MBO 算法的优化能力；Feng 等人[25]将混沌理论和高斯变异引入 MBO 算法，增强了 MBO 算法的全局优化性能；蒙丽萍等人[26]采用将种群动态随机分割成两个子群体的策略，以保持种群搜索的多样性；Feng 等人提出了一种多策略的 MBO 算法，以帮助 MBO 算法跳出局部最优点[27]；Feng 等人[28]将反向学习和高斯扰动引入 MBO 算法，从而加快 MBO 算法的收敛速度；Hu 等人[29]提出了一种自适应种群策略的 MBO 算法，提高了 MBO 算法的探索能力等。此外，MBO 算法也经常和其他优化算法进行结合。Ghanem 等人[30]将 MBO 算法和人工蜂群优化算法结合，以解决数值优化问题；Ghetas 等人[31]将和声搜索算法与 MBO 算法进行结合，提高了 MBO 算法的优化性能。然而，上述一些算法仍存在易陷入局部最优和收敛速度慢等问题，尤其是在解决一些复杂优化问题时，仍存在很大的改进空间。

本文针对 MBO 算法易陷入局部最优和收敛速度慢等问题，提出了一种基于改进的交叉迁移和共享调整的新的 MBO（CSMBO）算法。首先，针对 MBO 算法中的迁移算子，运用基于维度的垂直交叉操作替换原始迁移操作，形成新的交叉迁移算子，以增加种群多样性，提升其搜索能力，这样可以在一定程度上避免 MBO 算法陷入局部最优的问题；其次，将原始调整算子改为具有信息分享功能的共享调整算子，有效地利用种群的有效信息，进而加快 MBO 算法的收敛速度；最后，采用贪婪选择策略取代标准 MBO 算法中的精英保留策略，减少一次排序操作，有效提高了 MBO 算法的计算效率。为了验证 CSMBO 算法的优化能力，在一组不同维度的基准函数上进行测试实验，其结果表明 CSMBO 算法在解决复杂问题上具有良好的优化性能。

1　标准 MBO 算法

MBO 的相关知识可以参见文献[15, 24]。在 MBO 算法中，所有的蝴蝶个体都是理想化的，仅位于美国北部和加拿大南部（区域 1）、墨西哥（区域 2）两个区域。因此，蝴蝶的位置以迁移算子和调整算子两种方式更新。首先，后代的产生（位置更新）通过迁移算子；其次，其他蝴蝶的位置通过调整算子来更新。所以，蝴蝶个体的搜索方向是通过迁移算子和调整算子确定的，而且迁移算子和调整算子能够同时执行。因此，MBO 算法非常适合并行处理，并且能很好地平衡局部搜索能力和全局探索能力。MBO 算法遵循如下理想规则：

a）所有的蝴蝶都只在区域 1 和 2 中。即区域 1 和 2 中的蝴蝶组成了整个蝴蝶种群。

b)每一只蝴蝶后代都是区域 1 和 2 中的蝴蝶通过迁移算子产生的。

c）为了保持种群数量不变，一旦有后代蝴蝶产生，就会有相应的父代蝴蝶消失。

d）具有最好的适应度值的蝴蝶个体自动进入下一代，不对其进行任何操作。这样可以保证蝴蝶种群的质量不会随着迭代次数的增加而下降。

MBO 算法的两个位置更新策略如下：

a) 迁移算子。其目的是更新蝴蝶在区域 1 和 2 之间的迁徙。首先，蝴蝶总数为 NP，区域 1 和 2 中的蝴蝶数分别为 $NP1 = \mathrm{ceil}(p \times NP)$ 和 $NP2 = NP - NP1$，其中 p 为蝴蝶比例（迁移率），MBO 算法设置 $p = \frac{5}{12}$；$\mathrm{ceil}(x)$是舍入到大于或等于 x 的最接近的整数。区域 1 的子群记为 Subpopulation1，区域 2 的子群记为 Subpopulation2。 因此，迁移算子可以表示为

$$x_{i,k}^{t+1} = \begin{cases} x_{r_1,k}^{t}, r \le p \\ x_{r_2,k}^{t}, r > p \end{cases} \tag{1}$$

其中：$x_{i,k}^{t+1}$ 为 x_i 在 $t+1$ 时刻的第 k 维；$x_{r_1,k}^{t}$ 表示 x_{r_1} 在 t 时刻的第 k 维；$x_{r_2,k}^{t}$ 为 x_{r_2} 在 t 时刻的第 k 维。当前迭代次数由 t 表示。蝴蝶 r_1 和 r_2 分别从 Subpopulation1 和 Subpopulation2 中随机选出。$r = rand \times peri$，其中 $peri$ 为迁移期，在 MBO 算法中取 $peri = 1.2$，$rand$ 是[0, 1]中的随机数。

b) 调整算子。其目的是更新 Subpopulation2 中蝴蝶的位置。对于蝴蝶 j 的所有维，若 $rand \le p$，则该个体进行位置更新的公式如下：

$$x_{j,k}^{t+1} = x_{best,k}^{t} \tag{2}$$

其中：$x_{j,k}^{t+1}$ 为 x_j 在 $t+1$ 时刻的第 k 维；$x_{best,k}^{t}$ 为 Land1 与 Land2 当中最好个体的位置；$x_{best,k}^{t}$ 为 x_{best} 的第 k 维；t 为当前迭代次数。

若 $rand > p$，则该个体进行位置更新的公式如下：

$$x_{j,k}^{t+1} = x_{r_3,k}^{t} \tag{3}$$

其中：$x_{r_3,k}^{t}$ 为 x_{r_3} 的第 k 维；x_{r_3} 是从 Land2 中随机选出。在此条件下，若 $rand > BAR$，则该个体进行位置更新的公式如下：

$$x_{j,k}^{t+1} = x_{i,k}^{t+1} + \alpha(dx_k - 0.5) \tag{4}$$

其中：BAR 为调整率，在 MBO 算法中 $BAR = p$。如果 BAR 小于随机数 $rand$，在 $t+1$ 时刻的第 k 维就会更新，其中权重因子 α 的计算方式为 $\alpha = \frac{S_{\max}}{t^2}$，$S_{max}$ 为蝴蝶的最大步长，dx 为蝴蝶 j 的步长，通过 Levy flight 计算 $dx = Levy(x_j^t)$。

2　基于交叉迁移和共享调整的蝴蝶优化算法

2.1　改进算法的动机

MBO 算法的迁移算子采用式(1)所示的直接取代式迁移操作，虽然提供了一定的局部搜索能力，但迁移方式单一，可搜索位置局限，故将交叉操作与原迁移算子有机融合，形成交叉迁移算子，这样可以有效增加种群的多样性，提升算法的搜索能力。

从式(3)可以看出，第 $t+1$ 代蝴蝶的信息是直接从第 t 代继承的，于是可知这种直接继承的方法过于简单，没有充分利用种群的有效信息。因此，将共享操作与原调整算子结合来形成新的共享调整算子，这样可以有效地利用种群的优秀信息，有利于引导蝴蝶向最优解的方向靠近，进而加快算法的收敛速度。

标准 MBO 算法的精英保留策略需要设置参数和排序操作，增加了算法计算的复杂程度。因此，用贪婪选择策略替换精英保留策略，不仅可以避免精英参数的设置，而且还可以减少一次排序操作，进而降低算法的计算复杂度。

2.2　交叉迁移算子

鉴于 MBO 算法的迁移算子搜索能力不足，运用基于维度的垂直交叉操作取代其迁移算子的直接取代式迁移操作，形成新的交叉迁移算子，可以有效弥补原始迁移操作迁移形式简单、方向单一以及在解空间中可搜索到的位置较为局限

的不足。下面给出新的交叉迁移算子，其计算公式可以表示为：

$$x_{i,k}^{t+1}=\begin{cases}c\times x_{r_1,k}^{t}+(1-c)\times x_{r_1,q}^{t}, r\le p\\ c\times x_{r_2,k}^{t}+(1-c)\times x_{r_2,q}^{t}, r>p\end{cases} \tag{5}$$

其中：垂直交叉缩放因子 c = rand，q = rand × D。

对式(1)和式(5)的分析可知，在式(5)中，第 $t+1$ 代蝴蝶的第 k 维会受到第 t 代蝴蝶的第 k 维和第 q 维（q 是随机选择的）两者的共同影响，其具体方式是对两者进行加权交叉计算，这样可以增加种群的多样性，进而增强算法的搜索能力，有效克服式(1)存在的形式简单、方向单一等缺陷。

另外，由于贪婪选择策略具有只接受更优秀个体的优点，接下来可以将原迁移算子中的精英保留策略更换为贪婪选择策略，进而降低交叉迁移算子的计算复杂度。其表示公式如下：

$$x_{i,\mathrm{new}}^{t+1}=\begin{cases}x_i^{t+1}, f(x_i^{t+1})<f(x_i^{t})\\ x_i^{t}, \mathrm{otherwise}\end{cases} \tag{6}$$

其中：$x_{i,\mathrm{new}}^{t+1}$ 是下一代新生成的蝴蝶个体；$f(x_i^{t+1})$ 和 $f(x_i^{t})$ 分别表示蝴蝶 x_i^{t+1} 和 x_i^{t} 的适应度值。

下面给出交叉迁移算子的伪代码，如算法 1 所示。

算法 1 交叉迁移算子

for i = 1 to $NP1$ **do**
　for k = 1 to D **do**
　　根据式(5)更新 $x_{i,k}^{t+1}$
　end for k
　根据式(6)计算 $x_{i,\mathrm{new}}^{t+1}$
end for i

2.3 共享调整算子

针对调整算子中式(3)直接继承方式过于简单的缺点，为了充分利用种群中的有效信息，可以将共享操作合并到 MBO 算法的调整算子中，进而形成新的共享调整算子。其表示公式如下：

$$x_{j,k}^{t+1}=x_{best,k}^{t}+(rand-0.5)\times(x_{best,k}^{t}-x_{r_3,k}^{t}) \tag{7}$$

其中：$x_{best,k}^{t}$ 为种群中最好的个体 x_{best} 在 t 时刻的第 k 维；$rand$ 是[0, 1]中的随机数。

从式(7)可以看出，共享调整算子充分利用种群的有效信息，有利于引导蝴蝶向最优解的方向靠近，加快算法的收敛速度；同时，也将原始调整算子中的精英保留策略改为贪婪选择方案来提高共享调整算子的计算效率，其表示公式如式(6)所示。

下面给出共享调整算子的伪代码，如算法 2 所示。

算法 2 共享调整算子

for j = 1 to $NP2$ **do**
　for k = 1 to D **do**
　　取一个[0, 1]区间的随机数 $rand$
　　if $rand \le p$ then
　　　根据式(2)计算 $x_{j,k}^{t+1}$
　　else
　　　从子群 2（Subpopulation2）中随机选出一只蝴蝶（记作 r_3）
　　　根据式(3)计算 $x_{i,k}^{t+1}$
　　　if rand >BAR then
　　　　根据式(4)计算 $x_{i,k}^{t+1}$
　　　end if
　　end if
　end for k
　根据式(6)计算 $x_{i,\mathrm{new}}^{t+1}$
end for j

2.4 CSMBO 算法总流程

在提出的交叉迁移算子和共享调整算子的基础上，给出 CSMBO 算法的流程如图 1 所示。

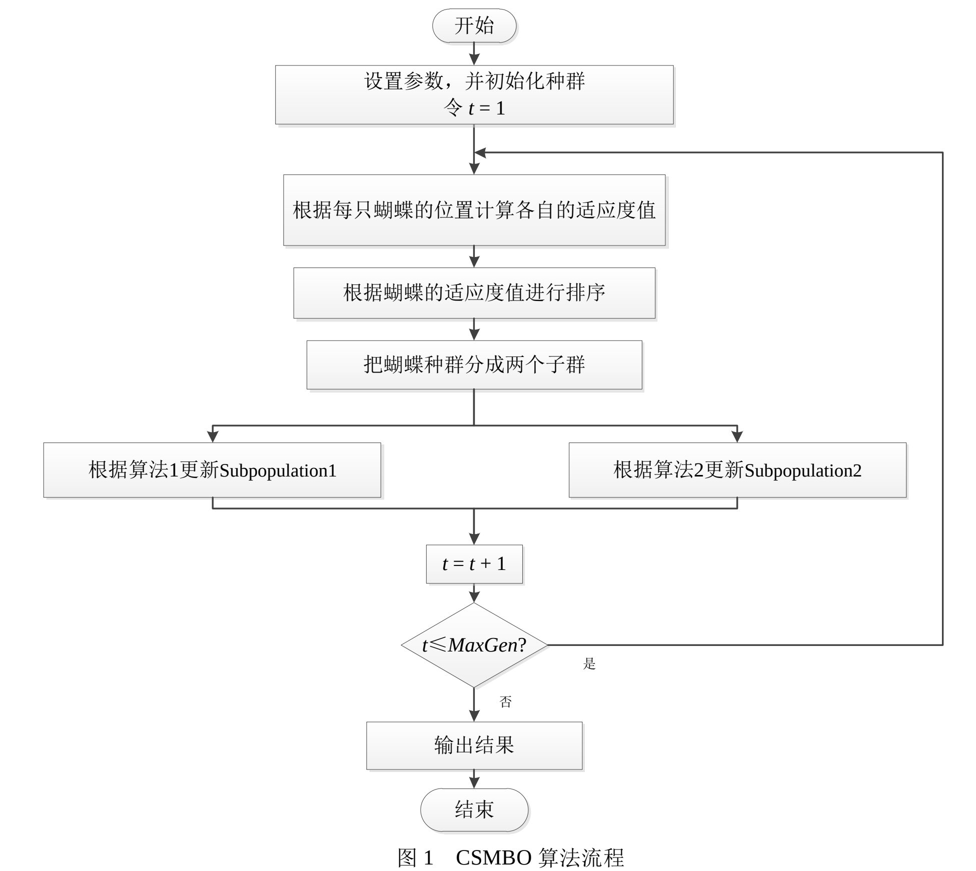


图 1　CSMBO 算法流程

Fig. 1　Main procedure of CSMBO algorithm

3 仿真实验

3.1 测试函数与参数设置

为了测试本文提出的 CSMBO 算法的有效性，在一组不同维度的基准函数上进行四种算法的优化实验对比。表 1 中给出了八个测试函数的信息。它们包括单峰函数如 f_1～f_5，多峰函数如 f_6～f_8。其中，单峰函数只有一个全局最优点，适合用于评估算法的搜索能力；多峰函数有多个局部最优点，适合用于评估算法的探索能力。F_{min} 是函数的理论最优值。本文所有仿真实验均在操作系统为 Windows 7、CPU 为主频 3.10 Ghz 和内存为 4 GB 的 PC 上实施，编程语言采用 MATLAB R2014a。

在实验中，将本文提出的 CSMBO 算法与其他三种相关的优化算法（PSO[9]、MBO[15]、GCMBO[24]）进行优化性能对比分析。本文用四种算法对不同维度的基准函数进行优化实验，以便验证 CSMBO 算法的有效性、高优化效率和普适性等。函数的维度选择包括 30 维和 50 维。不同的维度选择可以增加寻优难度，以便验证 CSMBO 算法处理各种复杂优化问题的能力。四种算法的共同参数设置如表 2 所示。其中，最大迭代次数 *MaxGen* 随优化函数的维数增加而增加，其他参数设置均与相关文献的 PSO、MBO、GCMBO 保持一致。在表 2 中，选择两个维度（30 维和 50 维）进行算法性能测试，其中 30 维对应的种群数量和最大迭代次数分别是 50 和 1 000，50 维对应的种群数量和最大迭代次数分别是 50 和 2 000。在实验实施过程中，若种群规模过小，会导致算法收敛速度过慢；若种群规模过大，会增加算法的时间复杂度，浪费不必要的存储空间。在文献[24]提出的结合贪婪策略和自适应交叉算子的 MBO 算法、文献[26]提出的采用动态分割种群策略的改进 MBO 算法以及文献[29]提出的基于自适应种群的改进 MBO 算法中，均将种群规模设置为 50，因此，本文在分配种群规模时，考虑到基准函数的维度和 MBO 算法的特点，也将种群规模设置为 50。为了避免随机性对实验结果的影响，在仿真实验中，四种算法对每一个优化问题均独立进行 30 次实验。

表 1 基准函数信息

Table 1 Information of benchmark functions

函数	公式	定义域	F_{min}
Quartic	$f_1(x)=\sum_{i=1}^{d} i x_i^4 + \text{random}[0,1)$	$[-1.28,1.28]^D$	0
Schwefel 1.2	$f_2(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^D$	0
Schwefel 2.21	$f_3(x)=\max_i(\lvert x_i\rvert, 1\le i\le d)$	$[-100, 100]^D$	0
Step	$f_4(x)=\sum_{i=1}^{d}(\lfloor x_i+0.5\rfloor)^2$	$[-100,100]^D$	0
Rosenbrock	$f_5(x)=\sum_{i=1}^{d-1}((x_i-1)^2+100(x_i^2-x_{i+1})^2)$	$[-10, 10]^D$	0
Penalized 1	$f_6(x)=\frac{\pi}{d}(10\sin^2(\pi y_i)+\sum_{i=1}^{d-1}(y_i-1)^2(1+10\sin^2(\pi y_{i+1}))+(y_d-1)^2)$ $+\sum_{i=1}^{d}u(x_i,10,100,4)$, 其中，$y_i=1+\frac{1}{4}(x_i+1)$，$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, x_i>a,\\ 0, -a\le x_i\le a,\\ k(-x_i-a)^m, x_i<-a,\end{cases}$	$[-50, 50]^D$	0
Penalized 2	$f_7(x)=0.1(\sin^2(\pi x_1)+\sum_{i=1}^{d-1}(x_i-1)^2(1+\sin^2(3\pi x_{i+1}))$ $(x_d-1)(1+\sin^2(2\pi x_d)))+\sum_{i=1}^{d}u(x_i,5,100,4)$	$[-50, 50]^D$	0
Griewank	$f_8(x)=\frac{1}{4000}\sum_{i=1}^{d}x_i^2-\prod_{i=1}^{d}\cos(\frac{x_i}{\sqrt{i}})+1$	$[-600, 600]^D$	0

表 2 四种算法的参数设置

Table 2 Values of parameters of four algorithms

种群数量	维度	最大迭代次数
50	30	1000
50	50	2000

3.2 优化性能比较

在 8 个基准函数上对不同维度（30 维和 50 维）的测试结果如表 3 和 4 所示。其中包含 30 次独立运行寻优结果中的最优值、最差值、平均值和标准差，粗体表示测试结果的最优者。文献[9]、文献[15]和文献[24]均指出平均值和标准差值越小说明算法的搜索能力越好且稳定性越强。

表 3 给出了四种算法在 30 维函数上的测试结果。从表 3 中的数值可以看出，在 f_1 和 f_5～f_7 上，GCMBO 算法在最优值上是最好的，但是 CSMBO 算法的最差值、平均值和标准差在四种算法中都是最优的。值得一提的是，在 f_5 和 f_6 上，CSMBO 算法的平均值和标准差明显比其他三种算法的结果更好。在 f_2 和 f_3 上，CSMBO 算法的优化结果均比其他三种算法优秀，特别是在 f_2 上，CSMBO 算法的结果大幅领先于其他三种算法。在 f_4 和 f_8 上，MBO 算法、GCMBO 算法和 CSMBO 算法在最优值的寻优上均取得了一致的结果，尤其是在 f_4 上，都找到了函数的理论最优值，但是 CSMBO 算法的最差值、平均值和标准差的结果都是最好的。总之，从 30 维函数上的实验结果可知，本文提出的 CSMBO 算法的优化性能相对优异。

表 4 给出了四种算法在 50 维函数上的测试结果。从表 4 中可以看出，对于 f_1 和 f_3，CSMBO 算法的寻优结果均是最优的。在 f_2 上，PSO 算法的最差值和标准差是最好的，而

CSMBO 算法的最优值和平均值是最优的。MBO 算法、GCMBO 算法和 CSMBO 算法均找到了 f_4 的理论最优值，但 CSMBO 算法的最差值、平均值和标准差结果都大幅优于其他三种算法。对于 f_5、f_6 和 f_7，GCMBO 算法的最优值结果是最好的，而 CSMBO 算法在最差值、平均值和标准差的评价指标上是最优秀的。值得一提的是，在 f_5 和 f_6 上，CSMBO 算法的结果都大幅度优于其他三种算法。总之，从 50 维函数上的实验结果可知，本文提出的 CSMBO 算法的优化性能是相对优秀的。

表 3 四种算法在 30 维函数上的测试结果

Table 3 Results of four algorithms on 30-dimensional functions

函数	算法	最优值	最差值	平均值	标准差
f_1	PSO	1.6029	12.6050	6.9126	2.5498
	MBO	0.0015	122.1438	50.5840	45.3491
	GCMBO	**3.3127e-24**	7.7247	1.5967	2.2949
	CSMBO	4.5748e-20	**1.6384**	**0.0629**	**0.2998**
f_2	PSO	8.8927e+03	1.8163e+04	1.3409e+04	2.5485e+03
	MBO	7.1783	6.5083e+04	2.4090e+04	1.8531e+04
	GCMBO	3.8183	2.6657e+04	1.1791e+04	8.5563e+03
	CSMBO	**4.3512e-08**	**0.0015**	**6.3360e-05**	**2.7170e-04**
f_3	PSO	37.4256	54.7068	46.3699	4.5268
	MBO	0.0010	87.6419	33.0509	22.3718
	GCMBO	0.0947	73.3843	31.8467	23.9338
	CSMBO	**6.6164e-05**	**12.2039**	**2.4087**	**3.5827**
f_4	PSO	10379	16015	1.3538e+04	1.4832e+03
	MBO	**0**	72991	2.5781e+04	2.7767e+04
	GCMBO	**0**	2206	203.7667	541.4205
	CSMBO	**0**	**7**	**0.2333**	**1.2780**
f_5	PSO	2.3207e+06	2.3380e+07	9.0459e+06	5.2705e+06
	MBO	1.2839e-11	5.8334e+08	1.3545e+08	1.9159e+08
	GCMBO	**2.5213e-16**	8.2390e+06	5.6040e+05	2.0417e+06
	CSMBO	6.9181e-12	**11.6346**	**0.8624**	**2.3041**
f_6	PSO	7.5324e+06	8.1079e+07	4.2706e+07	1.9612e+07
	MBO	9.8230e-10	1.0516e+09	1.4534e+08	2.6536e+08
	GCMBO	**4.7593e-14**	5.2262e+07	3.4594e+06	1.1075e+07
	CSMBO	8.1086e-11	**1.2936**	**0.0914**	**0.2833**
f_7	PSO	412.4324	1.5276e+03	963.1479	230.7740
	MBO	3.9119e-04	9.5470e+03	1.8755e+03	3.1023e+03
	GCMBO	**1.4106e-10**	626.1261	98.7457	142.3047
	CSMBO	1.7249e-05	**140.8104**	**52.2129**	**38.1776**
f_8	PSO	91.0693	142.6368	114.4050	12.9686
	MBO	**1.0000**	595.4902	148.0689	176.6548
	GCMBO	**1.0000**	130.8612	42.5119	45.7053
	CSMBO	**1.0000**	**1.0000**	**1.0000**	**2.5766e-11**

以上在不同维度基准函数上的实验结果证明了 CSMBO 算法的有效性与可行性，不仅在不同维度单峰函数上有较好的收敛精度，而且在不同维度多峰函数上显示了更好的全局搜索能力，以相对明显的优势优于其他三种对比算法。

3.3 收敛性分析

为了更直观地看出 CSMBO 算法的收敛速度、局部搜索能力和全局搜索能力，本文给出了四种算法在求解 50 维测试函数时的收敛曲线图，如图 2 所示。

从图 2 可以看出，CSMBO 算法在对八个测试函数进行优化时的收敛速度均优于其他三种对比算法。特别是在 f_3、f_6、f_7 和 f_8 上，CSMBO 算法的收敛速度大幅度优于对比算法。总之，从整体情况来看，CSMBO 算法在八个测试函数上的收敛性能都是最优的，再一次证明 CSMBO 算法的改进策略是有效的。其原因是由于 CSMBO 算法融入了交叉操作，增强了种群的多样性，提升了算法的全局勘探能力；而且在调整算子中加入了共享操作，种群中更优个体的信息得到有效利用，使得 CSMBO 算法的收敛速度得到了很大幅度的提升。

综上所述，CSMBO 算法不管在单模态函数上，还是在多模态函数上，与其他三种算法相比具有更优秀的优化性能，所以本文 CSMBO 算法的改进是有效和可行的。

表 4 四种算法在 50 维函数上的测试结果

Table 4 Results of four algorithms on 50-dimensional functions

函数	算法	最优值	最差值	平均值	标准差
f_1	PSO	21.1397	57.5429	39.4603	8.8743
	MBO	0.0034	415.5476	153.3220	142.7727
	GCMBO	1.9171e-07	133.2460	19.750	29.8177
	CSMBO	**1.8955e-17**	**0.6991**	**0.0437**	**0.1377**
f_2	PSO	2.4468e+04	**4.7567e+04**	3.6691e+04	**5.6002e+03**
	MBO	0.0010	1.6716e+05	7.1776e+04	5.1240e+04
	GCMBO	2.3034e+03	7.4799e+04	2.8123e+04	1.7575e+04
	CSMBO	**1.2244e-07**	5.7843e+04	**5.2787e+03**	1.5786e+04
f_3	PSO	56.5356	94.9000	87.4604	11.0036
	MBO	0.0071	90.5485	35.3305	26.7642
	GCMBO	0.0299	85.3000	33.9794	21.5405
	CSMBO	**5.2411e-05**	**15.5230**	**4.1900**	**4.9262**
f_4	PSO	26696	33563	3.0441e+04	1.9358e+03
	MBO	**0**	132209	5.1815e+04	5.1859e+04
	GCMBO	**0**	12346	1696	2.9548e+03
	CSMBO	**0**	**19**	**0.8333**	**3.6016**
f_5	PSO	1.9350e+07	8.9184e+07	4.9437e+07	1.8547e+07
	MBO	3.4589e-11	1.1141e+09	2.4720e+08	4.0804e+08
	GCMBO	**2.0184e-17**	5.4740e+07	3.2179e+06	1.0925e+07
	CSMBO	1.0028e-11	**41.5839**	**4.7557**	**9.4187**
f_6	PSO	8.6540e+07	2.4986e+08	1.5685e+08	4.5239e+07
	MBO	6.5719e-10	1.9497e+09	3.6131e+08	6.4416e+08
	GCMBO	**9.8724e-16**	1.4513e+08	1.0977e+07	3.1121e+07
	CSMBO	2.3708e-10	**194.3978**	**10.7529**	**37.6554**
f_7	PSO	1.3892e+03	3.5147e+03	2.4754e+03	478.7173
	MBO	1.3455e-04	1.8360e+04	5.9326e+03	6.2480e+03
	GCMBO	**3.3925e-09**	1.4881e+03	243.2459	353.8553
	CSMBO	1.4367e-05	**265.6663**	**65.0644**	**58.8563**
f_8	PSO	200.9274	306.9407	266.5297	26.2755
	MBO	1.2936	1.0722e+03	419.0091	381.3554
	GCMBO	**1.0000**	312.7389	96.2010	97.1711
	CSMBO	**1.0000**	**1.0000**	**1.0000**	**2.2857e-11**

3.4 CSMBO 算法的复杂度分析

在软硬件运行环境相同的前提下，优化算法的复杂度计算主要由目标函数的复杂度和算法自身流程的复杂度两部分组成。本文在对比实验中，设置所有算法（PSO、MBO、GCMBO 和本文的 CSMBO 算法）具有相同的种群数量和最大迭代次数，使得它们的最大函数评价次数（MNFE）[32]近似相等，故 CSMBO 算法的复杂度计算主要取决于自身流程的复杂程度。假设 CSMBO 算法的最大迭代次数为 T，种群规模为 N，子群 1 为 $N1$，子群 2 为 $N-N1$，维度为 D。由 CSMBO 算法流程（图 1）可知，该算法的时间复杂度主要由每次迭代循环决定，详细步骤分析如下：a)需要计算蝴蝶的

适应度值，时间复杂度为 $O(N)$；b)排序，时间复杂度是 $O(N^2)$；c)分成两个子群，时间复杂度是 $O(N)$；d)运行交叉迁移算子，有两个内循环，时间复杂度为 $O(N1 \times D)$，且在共享调整算子中，也存在两个内循环，时间复杂度为 $O((N-N1) \times D)$。所以，CSMBO 算法的总时间复杂度是 $T(n) = O(f(n)) = O(T \times (N + N^2 + N + (N1 \times D) + ((N-N1) \times D))) = O(T \times (2N + N^2 + N \times D)) = O(T \times N^2)$。在 CSMBO 算法中，由于可变的储存空间受到种群规模 N 和变量维度 D 的影响，则计算其空间复杂度为 $S(n) = O(f(n)) = O(N \times D)$。

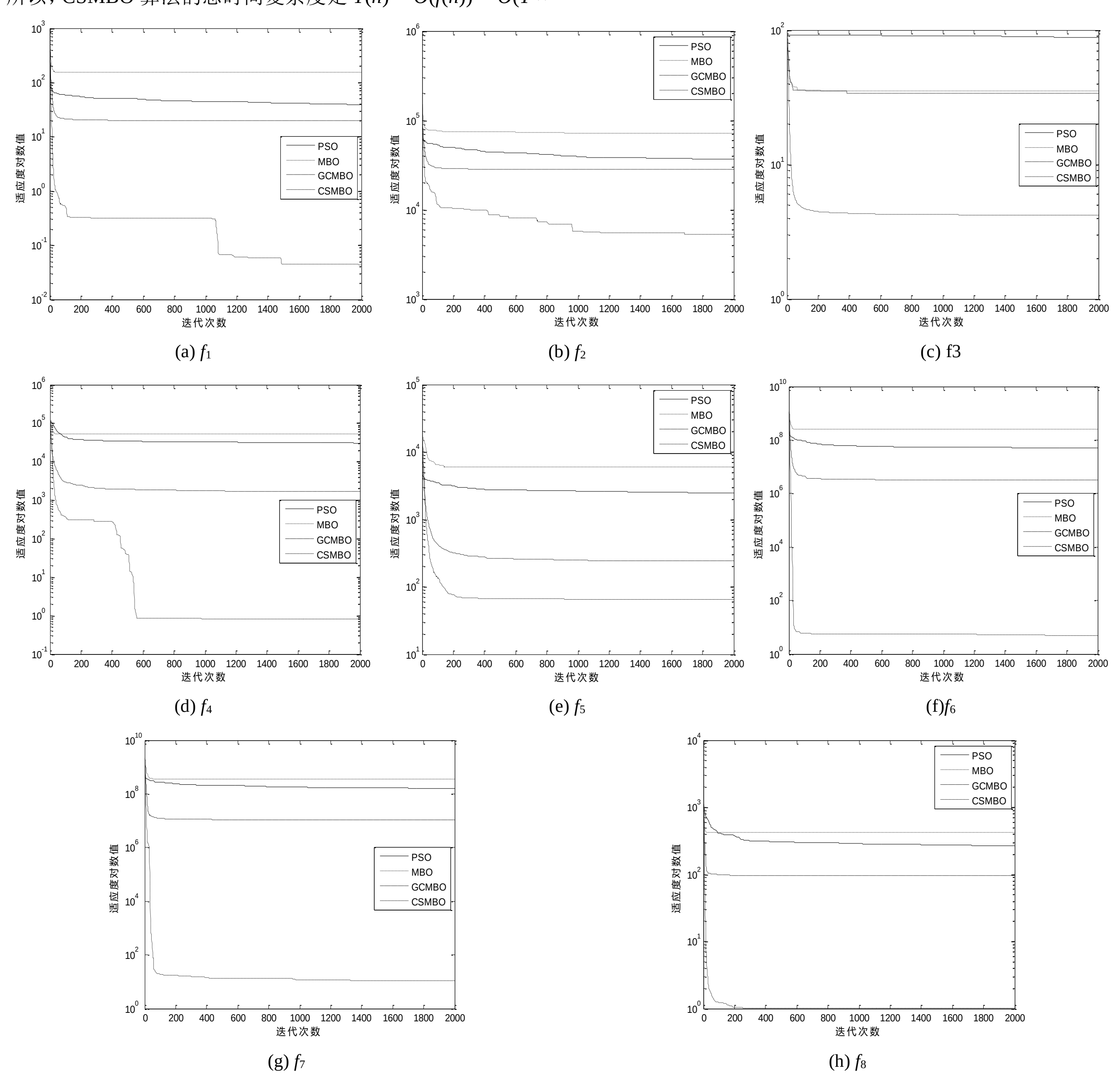


图 2 四种算法在八个基准函数上的收敛曲线

Fig. 2 Convergence curves of four algorithms on eight benchmark functions

4 结束语

本文针对 MBO 算法易陷入局部最优和收敛速度慢等问题，提出了一种基于交叉迁移和共享调整的改进 MBO（CSMBO）算法。首先，将 MBO 的迁移算子改为基于维度垂直交叉操作的交叉迁移算子，克服了原始迁移算子方式单一的缺陷，而且增加了种群的多样性，这在一定程度上避免了 MBO 算法陷入局部最优；其次，用具有信息分享功能的共享调整算子替换原始调整算子，加快了 MBO 算法的收敛速度；最后，将 MBO 算法中的精英保留策略改为贪婪选择策略，减少了精英参数的设置。在一组多维度的基准函数上的实验证明，与其他三种对比算法相比，本文提出的 CSMBO 算法在解决函数优化问题时具有更好的全局探索能力和局部搜索能力。

参考文献：

[1] 盛四清，陈玉良，张晶晶．基于差分进化人工蜂群算法的光伏最大功率跟踪策略研究 [J]．电力系统保护与控制，2018, 46 (11): 23-29. (Sheng Siqing, Chen Yuliang, Zhang Jingjing. Research on maximum power point tracking strategy based on differential evolution artificial bee colony algorithm of photovoltaic system [J]. Power System Protection and Control, 2018, 46 (11): 23-29.)

[2] Rakhshani H, Rahati A. Snap-drift cuckoo search: a novel cuckoo search optimization algorithm [J]. Applied Soft Computing, 2017, 52 (3): 771-794.

[3] 康健，靳斌，段秀娟，等．基于贝叶斯—粒子群算法的微电网优化运

行 [J]. 电力系统保护与控制，2018, 46 (12): 32-41. (Kang Jian, Jin Bin, Duan Xiujuan, *et al.* Optimal operation of microgrid based on Bayesian-PSO algorithm [J]. Power System Protection and Control, 2018, 46 (12): 32-41.)

[4] 江岳春，何钟南，刘爱玲. 基于改进 BBO 算法的风电—水电互补优化运行策略 [J]. 电力系统保护与控制, 2018, 46 (10): 39-47. (Jiang Yuechun, He Zhongnan, Liu Ailing. A complementary optimal operation strategy of wind power-hydropower based on improved biogeography-based optimization algorithm [J]. Power System Protection and Control, 2018, 46 (10): 39-47.)

[5] 王静，李维德. 基于 CEEMD 和 GWO 的超短期风速预测 [J]. 电力系统保护与控制，2018, 46 (9): 69-74. (Wang Jing, Li Weide. Ultra-short-term forecasting of wind speed based on CEEMD and GWO [J]. Power System Protection and Control, 2018, 46 (9): 69-74.)

[6] Wang Feng, Zhang Heng, Li Kangshun, *et al.* A hybrid particle swarm optimization algorithm using adaptive learning strategy [J]. Information Sciences, 2018, 436: 162-177.

[7] 刘沛，高岳林，郭伟. 一种基于改进的磷虾群和粒子群的混合算法 [J]. 河南师范大学学报：自然科学版, 2017, 45 (2): 119-124. (Liu Pei, Gao Yuelin, Guo Wei. A hybrid algorithm based on improved krill herd and particle swarm optimization [J]. Journal of Henan Normal University: Natural Science Edition, 2017, 45 (2): 119-124.)

[8] 袁晗，徐春梅，杨平，等. 一种基于子群变异的粒子群优化算法 [J]. 计算机应用研究, 2017, 34 (4): 1076-1079. (Yuan Han, Xu Chunmei, Yang Ping, *et al.* New particle swarm optimization based on subswarm mutation [J]. Application Research of Computers, 2017, 34 (4): 1076-1079.)

[9] Kennedy J. Particle swarm optimization [M]// Encyclopedia of Machine Learning. Boston, MA: Springer, 2011: 760-766.

[10] 史旭栋，高岳林，韩俊茹. 基于模糊推理的粒子群优化算法 [J]. 河南师范大学学报：自然科学版，2017, 45 (2): 108-118. (Shi Xudong, Gao Yuelin, Han Junrun. Particle swarm optimization algorithm based on fuzzy reason [J]. Journal of Henan Normal University: Natural Science Edition, 2017, 45 (2): 108-118.)

[11] 张新明，王霞，涂强，等. 融合榜样学习和反向学习的粒子群优化算法 [J]. 河南师范大学学报：自然科学版, 2017, 45 (6): 91-99. (Zhang Xinming, Wang Xia, Tu Qiang, *et al.* Particle swarm optimization algorithm based on combining example learning and opposition learning [J]. Journal of Henan Normal University: Natural Science Edition, 2017, 45 (6): 91-99.)

[12] 张俊武，王德林，陈斌，等. 基于 PSO-GSA 算法的含 DFIG 互联系统 AGC 优化控制研究 [J]. 电力系统保护与控制, 2018, 46 (13): 48-54. (Zhang Junwu, Wang Delin, Chen Bin, *et al.* Research on PSO-GSA algorithm optimization for interconnected AGC system including DFIG wind turbines [J]. Power System Protection and Control, 2018, 46 (13): 48-54.)

[13] 刘小垒，张小松，蒋义琪，等. 基于分布式烟花算法的 WSN 布局优化方法 [J]. 计算机应用研究，2018, 35 (2): 569-572. (Liu Xiaolei, Zhang Xiaosong, Jiang Yiqi, *et al.* Deployment optimization of wireless sensor network based on parallelized fireworks algorithm [J]. Application Research of Computers, 2018, 35 (2): 569-572.)

[14] Zhang Xinming, Kang Qiang, Tu Qiang, *et al.* Efficient and merged biogeography-based optimization algorithm for global optimization problems [J/OL]. (2018).https://doi. org/10. 1007/s00500-018-3113-1.

[15] Wang Gaige, Deb S, Cui Zhihua. Monarch butterfly optimization [C]// Proc of Neural Computing and Applications. 2015: 1-20.

[16] 陈小雪，尉永清，任敏，等. 基于萤火虫优化的加权 K-means 算法 [J]. 计算机应用研究，2018, 35 (2): 466-470. (Chen Xiaoxue, Wei Yongqing, Ren Min, *et al.* Weighted K-means clustering algorithm based on firefly algorithm [J]. Application Research of Computers, 2018, 35 (2): 466-470.)

[17] 胡博，王昕，郑益慧，等. 基于萤火虫优化算法的分布式发电设备容量分配及配电网孤岛划分 [J]. 电力系统保护与控制, 2018, 46 (13): 21-26. (Hu Bo, Wang Xin, Zheng Yihui, *et al.* Calculation of isolated island partition and distributed generator capacity based on firefly algorithm [J]. Power System Protection and Control, 2018, 46 (13): 21-26.)

[18] 许喆，潘金生，樊淑娴，等. 基于改进萤火虫算法的含 DG 配电网重构方法 [J]. 电力系统保护与控制, 2018, 46 (14): 26-32. (Xu Zhe, Pan Jinsheng, Fan Shuxian, *et al.* A distribution network reconfiguration method with distributed generation based on improved firefly algorithm [J]. Power System Protection and Control, 2018, 46 (14): 26-32.)

[19] Feng Yanhong, Wang Gaige, Deb S, *et al.* Solving 0-1 knapsack problem by a novel binary monarch butterfly optimization [J]. Neural Computing and Applications, 2017, 28 (7): 1619-1634.

[20] Yadav V, Ghoshal S P. Optimal power flow for IEEE 30 and 118-bus systems using monarch butterfly optimization [C]// Proc of IEEE International Conference on Technologies for Smart-City Energy Security and Power. Piscataway, NJ: IEEE Press, 2018: 1-6.

[21] Faris H, Aljarah I, Mirjalili S. Improved monarch butterfly optimization for unconstrained global search and neural network training [J]. Applied Intelligence, 2018, 48 (2): 445-464.

[22] Chen Shifeng, Chen Rong, Gao Jian. A monarch butterfly optimization for the dynamic vehicle routing problem [J]. Algorithms, 2017, 10 (3): 1-19.

[23] Devikanniga D, Joshua S R R. Classification of osteoporosis by artificial neural network based on monarch butterfly optimization algorithm [J]. Healthcare Technology Letters, 2018, 5 (2): 70-75.

[24] Wang Gaige, Zhao Xinchao, Deb S. A novel monarch butterfly optimization with greedy strategy and self-adaptive [C]// Proc of the 2nd IEEE International Conference on Soft Computing and Machine Intelligence. Piscataway, NJ: IEEE Press, 2015: 45-50.

[25] Feng Yanhong, Yang Juan, Wu Congcong, *et al.* Solving 0-1 knapsack problems by chaotic monarch butterfly optimization algorithm with Gaussian mutation [J]. Memetic Computing, 2018, 10 (2): 135-150.

[26] 蒙丽萍，王勇，黄华娟. 采用动态分割种群策略的改进 MBO [J]. 计算机工程与应用, 2017, 53 (18): 149-156. (Meng Liping, Wang Yong, Huang Huajuan. Improved monarch butterfly optimization by using strategy of dynamic-dividing population [J]. Computer Engineering and Applications, 2017, 53 (18): 149-156.)

[27] Feng Yanhong, Wang Gaige, Li Wenbin, *et al.* Multi-strategy monarch butterfly optimization algorithm for discounted{0-1}knapsack problem [J]. Neural Computing and Applications, 2017, 30 (10): 3019-3036.

[28] Feng Yanhong, Wang Gaige, Dong Junyu, *et al.* Opposition-based learning monarch butterfly optimization with Gaussian perturbation for large-scale 0-1 knapsack problem [J]. Computers and Electrical Engineering, 2018, 67: 454-468.

[29] Hu Hui, Cai Zhaoquan, Hu Song, *et al.* Improving monarch butterfly optimization algorithm with self-adaptive population [J]. Algorithms, 2018, 11 (5): Article ID: 71.

[30] Ghanem W A H M, Jantan A. Hybridizing artificial bee colony with monarch butterfly optimization for numerical optimization problems [J].

Neural Computing and Applications, 2018, 30 (1): 163-181.

[31] Ghetas M, Yong C H, Sumari P. Harmony-based monarch butterfly optimization algorithm [C]// Proc of IEEE International Conference on Control System, Computing and Engineering. Piscataway, NJ: IEEE Press, 2015: 156-161.

[32] 张新明，康强，王霞，等. 差分迁移和趋优变异的生物地理学优化算法 [J]. 小型微型计算机系统，2018, 39 (6): 1168-1177. (Zhang Xinming, Kang Qiang, Wang Xia, *et al.* Biogeography-based optimization with differential migration and global-best mutation [J]. Journal of Chinese Computer Systems, 2018, 39 (6): 1168-1177.)